FINAL
~~Annual~~ Report (2/1/94-1/31/95)

AFOSR Grant No. F49620-94-1-0156

DEVELOPMENT OF AN ADVANCED IMPLICIT ALGORITHM FOR MHD COMPUTATIONS ON PARALLEL SUPERCOMPUTERS

Submitted to

Air Force Office of Scientific Research
Bolling AF Base
Washington, DC 20332-0001

UNIVERSITY OF WASHINGTON
Aerospace & Energetics Research Laboratory, FL-10
Seattle, Washington 98195

submitted by

Dr. Uri Shumlak
Research Assistant Professor

March 28, 1995

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: unclassified/unlimited	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: unclassified/unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Aerospace & Energetics Research Program; 6b. OFFICE SYMBOL (If applicable): FL-10	7a. NAME OF MONITORING ORGANIZATION: Office of Naval Research (ONR)
6c. ADDRESS (City, State and ZIP Code): University of Washington, Grant & Contract Services, 3935 Univ Way NE, Seattle, WA 98105-6613	7b. ADDRESS (City, State and ZIP Code): 1107 NE 45th St., Suite 350 /JD-16, Seattle, WA 98105
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): /NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F08671-9400676 / F49620-94-1-0156
8c. ADDRESS (City, State and ZIP Code): 110 Duncan Avenue, Suite B115, Bolling AFB, DC 20332-0001	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO.: 61102F; PROJECT NO.: 2304 ~~A000~~; TASK NO.: CS ~~100~~; WORK UNIT NO:
11. TITLE (Include Security Classification): DEVELOPMENT OF AN ADVANCED IMPLICIT ALGORITHM FOR MHD COMPUTATIONS ON PARALLEL SUPERCOMPUTERS (unclassified/unlimited)	
12. PERSONAL AUTHOR(S): SHUMLAK, Uri	
13a. TYPE OF REPORT: ~~Final~~ annual report; 13b. TIME COVERED: FROM 94/2/1 TO 95/1/31	14. DATE OF REPORT (Yr., Mo., Day): 95/3/28; 15. PAGE COUNT: 30
16. SUPPLEMENTARY NOTATION	

17 COSATI CODES		
FIELD	GROUP	SUB. GR

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): implicit algorithm, parallel computer, lower-upper symmetric-Gauss-Seidel, LUSGS, approximate Riemann solver, magnetohydrodynamic, MHD, Hartmann flow

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The primary objective of this project is to develop an advanced algorithm for parallel supercomputers to model time-dependent magnetohydrodynamics (MHD) in all three dimensions. A production code with this algorithm will provide a valuable tool for the design and testing of plasma related technologies that are important to the Air Force and industry. Implementing the algorithm on parallel supercomputers will allow the detailed modeling of realistic plasmas in complex three-dimensional geometries.

The algorithm incorporates an approximate Riemann solver with explicit diffusive terms. The system of equations is solved using an implicit lower-upper symmetric-Gauss-Seidel (**LUSGS**) relaxation method.

A two-dimensional version of the algorithm has been developed, placed into a testbed code, modified to include viscous and resistive effects, and tested against known analytical problems. The algorithm has been benchmarked to the one-dimensional shock tube and two-dimensional fully-developed (unmagnetized) viscous flow and (magnetized) Hartmann flow. The algorithm has been implemented on a parallel architecture and parallelization strategies have been investigated.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: unclassified /unlimited
22a. NAME OF RESPONSIBLE INDIVIDUAL: Donald W. Allen, Director G&C Services	22b. TELEPHONE NUMBER (Include Area Code): 206 543-4043; 22c. OFFICE SYMBOL: 6613

Development of an Advanced Implicit Algorithm for MHD Computations on Parallel Supercomputers

Contract Number: F49620-94-1-0156

U. Shumlak
Department of Aeronautics and Astronautics, FL-10
University of Washington, Seattle, WA 98195

1 Executive Summary

The primary objective of this project is to develop an advanced algorithm for parallel supercomputers to model time-dependent magnetohydrodynamics (MHD) in all three dimensions. This will provide a valuable tool for the design and testing of plasma related technologies that are important to the Air Force and industry. Implementing the algorithm on parallel supercomputers will allow the detailed modeling of realistic plasmas in complex three-dimensional geometries.

We have developed a two-dimensional version of the algorithm, placed it into a testbed code, added the modifications necessary for viscous and resistive effects, and tested the code against known analytical problems. We have implemented the algorithm on a parallel architecture and investigated parallelization strategies. Future plans include adding the time-accurate feature, installing the algorithm into MACH2, optimizing the parallelization, extending the code to three dimensions, and installing the three dimensional algorithm into MACH3.[1]

As a result of this project several professional collaborations now exist between the Department of Aeronautics and Astronautics at the University of Washington and the Air Force Phillips Laboratory, the University of Michigan, the University of Colorado, and other departments at the University of Washington. The work from this project has been presented at

an international conference and a publication will soon be submitted to a refereed journal.

2 Project Description

Plasmas are essential to many technologies that are important to the Air Force, some of which have dual-use potential. These applications include nuclear weapons effects simulations, radiation production for counter proliferation, fusion for power generation, and advanced plasma thrusters for space propulsion. In general, plasmas fall into a density regime where they exhibit both collective (fluid) behavior and individual (particle) behavior. Many plasmas of interest can be modeled by treating the plasma like a conducting fluid and assigning macroscopic parameters that accurately describe its particle-like interactions. The magnetohydrodynamic (MHD) model is a plasma model of this type.

2.1 Research Objectives

The objectives of the project are to:

- Develop a coupled, implicit, time-accurate algorithm for three-dimensional, viscous, resistive MHD simulations;
- Incorporate the algorithm into the MACH3 code, which was developed at the Air Force Phillips Laboratory;
- Validate the code with analytical and experimental data; and
- Apply the code to analyze plasma experiments at the University of Washington [Helicity Injected Tokamak (HIT)[2]] and at the Phillips Laboratory [the liner implosion system (WFX)[3] and the compact toroid experiment (MARAUDER)[4]].

2.2 Technical Description

2.2.1 MHD Plasma Model

The three-dimensional, viscous, resistive MHD plasma model is a set of mixed hyperbolic and parabolic equations. The Navier-Stokes equations are also of this type. This project applies some advances that have been made in implicit algorithms for the Navier-Stokes equations to the MHD equations.

These implicit algorithms solve the equation set in a fully coupled manner, which generates better accuracy than the current methods used for MHD simulations.

When expressed in conservative, non-dimensional form, the MHD model is described by the following equation set.

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho\mathbf{v}\\ \mathbf{B}\\ e\end{bmatrix}+\nabla\cdot\begin{bmatrix}\rho\mathbf{v}\\ \rho\mathbf{v}\mathbf{v}-\mathbf{B}\mathbf{B}+(p+\mathbf{B}\cdot\mathbf{B}/2)\,\bar{\bar{\mathrm{I}}}\\ \mathbf{v}\mathbf{B}-\mathbf{B}\mathbf{v}\\ (e+p+\mathbf{B}\cdot\mathbf{B}/2)\,\mathbf{v}-(\mathbf{B}\cdot\mathbf{v})\,\mathbf{B}\end{bmatrix}=$$

$$\nabla\cdot\begin{bmatrix}0\\ (ReAl)^{-1}\,\bar{\bar{\tau}}\\ (RmAl)^{-1}\,\bar{\bar{\eta}}\cdot\nabla\mathbf{B}\\ (ReAl)^{-1}\,\mathbf{v}\cdot\bar{\bar{\tau}}-(RmAl)^{-1}\,\bar{\bar{\eta}}\cdot(\nabla\times\mathbf{B})\times\mathbf{B}+\frac{M_i}{2}\,(PeAl)^{-1}\,\bar{\bar{k}}\cdot\nabla T\end{bmatrix} \quad (1)$$

The variables are density (ρ), velocity ($\mathbf{v}$), magnetic induction ($\mathbf{B}$), pressure (p), energy density (e), and temperature (T). M_i is the ion mass. The energy density is

$$e=\frac{p}{\gamma-1}+\rho\frac{\mathbf{v}\cdot\mathbf{v}}{2}+\frac{\mathbf{B}\cdot\mathbf{B}}{2} \quad (2)$$

where $\gamma = c_p/c_v$ is the ratio of the specific heats. The non-dimensional tensors are the stress tensor ($\bar{\bar{\tau}}$), the electrical resistivity ($\bar{\bar{\eta}}$), and the thermal conductivity ($\bar{\bar{k}}$), and $\bar{\bar{\mathrm{I}}}$ is the identity matrix. The non-dimensional numbers are defined as follows:

$$\begin{array}{lll}\text{Alfvén Number :} & Al & \equiv V_A/V\\ \text{Reynolds Number :} & Re & \equiv LV/\nu\\ \text{Magnetic Reynolds Number :} & Rm & \equiv \mu_o LV/\eta\\ \text{Péclet Number :} & Pe & \equiv LV/\kappa\end{array} \quad (3)$$

The characteristic variables are length (L), velocity (V), Alfvén speed ($V_A = B/\sqrt{\mu_o\rho}$), kinematic viscosity (ν), electrical resistivity (η), and thermal diffusivity ($\kappa = k/\rho c_p$). μ_o is the permeability of free space ($4\pi\times10^{-7}$).

For convenience, the MHD equation set [eqn(1)] is rewritten in the following compact form

$$\frac{\partial Q}{\partial t}+\nabla\cdot\bar{\bar{T}}_h=\nabla\cdot\bar{\bar{T}}_p, \quad (4)$$

where Q is the vector of conservative variables, $\bar{\bar{T}}_h$ is the tensor of hyperbolic fluxes, and $\bar{\bar{T}}_p$ is the tensor of parabolic fluxes. The forms of these vectors

and tensors can be seen from eqn(1). The hyperbolic fluxes are associated with wave-like motion, and the parabolic fluxes are associated with diffusion-like motion.

2.2.2 Algorithm Overview

Because of the natural differences between hyperbolic and parabolic equations, the methods for solving them are very different. Since the MHD equations are of mixed type the hyperbolic and parabolic terms must be handled differently. The hyperbolic fluxes are differenced by applying an implicit, approximate Riemann algorithm that properly accounts for their wave-like behavior. The parabolic terms are discretized by applying explicit central differencing.

The design of the overall algorithm is primarily driven by the numerical techniques that must be used to discretize the hyperbolic terms. Therefore, we begin by considering the ideal MHD equations, which are obtained from eqn(4) by setting all the parabolic terms ($\bar{\bar{T}}_p$) to zero.

In one dimension they are

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} = \frac{\partial Q}{\partial t} + A\frac{\partial Q}{\partial x} = 0, \tag{5}$$

where F is the flux vector in the x direction (i.e. $\bar{\bar{T}}_h = (F, G, H)$) and A is its Jacobian.

$$A = \frac{\partial F}{\partial Q} \tag{6}$$

Here, Q is the vector of conserved variables:

$$Q = (\rho, \rho v_x, \rho v_y, \rho v_z, B_y, B_z, e)^T . \tag{7}$$

This is a set of hyperbolic equations and thus A has a complete set of real eigenvalues given by

$$\lambda = (v_x, v_x \pm V_{fast}, v_x \pm V_{slow}, v_x \pm V_{Ax})^T , \tag{8}$$

where V_{fast} and V_{slow} are the fast and slow magnetosonic speeds, and V_{Ax} is the Alfven speed based on the x component of the magnetic field. These can be expressed as

$$V_{fast}^2 = \frac{1}{2}\left[c_s^2 + V_A^2 + \sqrt{\left(c_s^2 + V_A^2\right)^2 - 4c_s^2 V_{Ax}^2}\right], \tag{9}$$

$$V_{slow}^2 = \frac{1}{2}\left[c_s^2 + V_A^2 - \sqrt{\left(c_s^2 + V_A^2\right)^2 - 4c_s^2 V_{Ax}^2}\right], \tag{10}$$

$$V_{Ax}^2 = \frac{B_x^2}{\mu_o \rho}. \tag{11}$$

Here, c_s is the ion sound speed, which for a perfect gas is

$$c_s^2 = \frac{\gamma p}{\rho}. \tag{12}$$

Information propagates along characteristics which travel at wave speeds given by the eigenvalues. Most modern numerical techniques for solving hyperbolic equations are based upon the idea of splitting the fluxes into components due to left and right running waves. Then each part of the flux can be differenced in an upwind manner, which greatly reduces numerical oscillations and stabilizes the solutions.

It is well known that if a hyperbolic equation is solved with an explicit scheme, then the allowable time step to maintain numerical stability is given by the CFL (Courant-Friedrichs-Lewy) condition, which in the case of the 1D MHD equations is

$$\Delta t < \frac{\Delta x}{|v_x + V_{fast}|}. \tag{13}$$

For the high magnetic fields and low densities common in many plasma experiments, the fast magnetosonic speed is quite high, and thus the time step is prohibitively small. We are often interested in only modeling the physics that occurs slower than Alfvén time scales. For example, it can be shown that resistive tearing modes, which are important in studying fusion plasmas, evolve on a time scale that is given by[5]

$$\tau_{tearing} \propto \tau_A^{2/5} \tau_\eta^{3/5} = (Lu)^{3/5}\, \tau_A. \tag{14}$$

τ_A is the Alfvén time, τ_η is the resistive diffusion time, and Lu is the Lundquist number, which is given by

$$Lu = \frac{\tau_\eta}{\tau_A} = RmAl. \tag{15}$$

If Lu is 10^6, which is typical for laboratory plasmas in fusion applications, the resistive tearing time is approximately 4000 times larger than the Alfvén time. By treating the hyperbolic fluxes implicitly in time, the stability restriction on the time step is removed, and the solution can be advanced at

the larger resistive tearing time step. This is our motivation for proposing an implicit scheme.

The starting point for deriving the algorithm is the two-dimensional ideal MHD equations in Cartesian form

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0. \tag{16}$$

We then discretize eqn(16) using first order Euler time differencing to get

$$\frac{\left(Q_{ij}^{n+1} - Q_{ij}^{n}\right)}{\Delta t} = -R_{ij}\left(Q^{n+1}\right) = -R_{ij}^{n+1} \tag{17}$$

where R is

$$R_{ij} = F_{i+1/2,j} - F_{i-1/2,j} + G_{i,j+1/2} - G_{i,j-1/2}. \tag{18}$$

Note that in this equation and all that follow the grid metric terms (cell areas and volumes) are omitted for clarity. We linearize R as follows:

$$R_{ij}^{n+1} \approx R_{ij}^{n} + \left(\frac{\partial R}{\partial Q}\right)_{ij}^{n}\left(Q_{ij}^{n+1} - Q_{ij}^{n}\right) \tag{19}$$

Substituting this expression back into eqn(17) and rearranging, we get

$$\left[\frac{\bar{\bar{\mathrm{I}}}}{\Delta t} + \left(\frac{\partial R}{\partial Q}\right)_{ij}^{n}\right]\Delta Q_{ij}^{n} = -R_{ij}^{n}. \tag{20}$$

Here ΔQ is defined as

$$\Delta Q^{n} \equiv Q_{ij}^{n+1} - Q_{ij}^{n}. \tag{21}$$

The left hand side of the eqn(20) is an implicit operator operating on ΔQ. It is a large banded block matrix. In three dimensions, it is an $(I_{max} \times J_{max} \times K_{max})$ by $(I_{max} \times J_{max} \times K_{max})$ matrix, where I_{max} is the number of cells in the x direction, etc. It is quite costly to invert a matrix of this size directly. We choose to invert it using an approximate factorization, which can be done more efficiently. When solved this way, eqn(20) is no longer time accurate. However, we can still achieve time accuracy with this type of scheme by adding the time derivative of Q as a source term to the right hand side of the equation. We then have

$$\left(\frac{\partial Q}{\partial \tau}\right)^{n+1} = -R_{ij}^{n+1} - S_{ij}^{n+1} \tag{22}$$

where

$$S_{ij}^{n+1} = \frac{1}{2\Delta t}\left(3Q_{ij}^{n+1} - 4Q_{ij}^{n} + Q_{ij}^{n-1}\right) \approx \frac{\partial Q}{\partial t}. \tag{23}$$

The τ in eqn(22) can be thought of as a pseudo time variable. At each physical time step, eqn(22) is solved iteratively in pseudo time until the left hand side vanishes. When the solution converges, our original equation

$$\frac{\partial Q}{\partial t} = -R \tag{24}$$

is solved. This technique is known as dual time-stepping.[6] Note that in eqn(23) a more accurate time derivative can be used at the expense of the additional memory needed to store the Q vectors from previous time steps.

One advantage of the strategy outlined above is that the implicit operator and the right hand side in eqn(20) are decoupled. The structure of the matrix no longer depends on the details of the discretization of the right hand side fluxes. In the following sections we will describe the relaxation scheme that is used to iteratively invert the implicit operator and the approximate Riemann solver that is used to form the right hand side fluxes.

2.2.3 LU-SGS Relaxation Scheme

We use the lower-upper symmetric-Gauss-Seidel (LU-SGS) method to iteratively invert the implicit operator.[7] To derive this method, we first consider the following first order accurate flux-vector splitting of R (at time level $n+1$) in eqn(17):

$$R_{ij} = F_{ij}^{+} - F_{i-1,j}^{+} + F_{i+1,j}^{-} - F_{ij}^{-} + G_{ij}^{+} - G_{i,j-1}^{+} + G_{i,j+1}^{-} - G_{ij}^{-} \tag{25}$$

where F^+ is the portion of the F flux vector corresponding to right-running waves, and F^- is the portion corresponding to left-running waves, and G^+ and G^- are similarly defined. This equation is linearized to obtain

$$\left\{\bar{\bar{1}} + \Delta t\left(A_{ij}^{+} - A_{i-1,j}^{+} + A_{i+1,j}^{-} - A_{i,j}^{-} + B_{ij}^{+} - B_{i,j-1}^{+} + B_{i,j+1}^{-} - B_{i,j}^{-}\right)\right\} \times \Delta Q_{ij}^{n} = -\Delta t R_{ij}^{n} \tag{26}$$

where A^+ is the Jacobian of F^+, and so on. We approximate these Jacobians as

$$A^{+} = \frac{1}{2}\left(A + \rho_A\right) \tag{27}$$

$$A^{-} = \frac{1}{2}\left(A - \rho_A\right) \tag{28}$$

where ρ_A is the maximum eigenvalue (spectral radius) of A. If we then iteratively solve this simplified implicit operator using a forward Gauss-Seidel sweep followed by a backward sweep, the resulting algorithm can be written as

$$\begin{aligned}\left\{\bar{\bar{\mathrm{I}}} + \Delta t\left[(\rho_A + \rho_B)\bar{\bar{\mathrm{I}}} - A^{+}_{i-1,j} - B^{+}_{i,j-1}\right]\right\} \\ \times \left\{\bar{\bar{\mathrm{I}}} + \Delta t\left[(\rho_A + \rho_B)\bar{\bar{\mathrm{I}}} + A^{-}_{i+1,j} + B^{-}_{i,j+1}\right]\right\} \\ \times \Delta Q^{n}_{ij} = -\left[1 + \Delta t\,(\rho_A + \rho_B)\right]\Delta t R^{n}_{ij}\end{aligned} \tag{29}$$

The forward sweep is equivalent to inverting a lower block diagonal matrix [the first braced term in eqn(29)], and the backward sweep is equivalent to inverting an upper block diagonal matrix [second braced term in eqn(29)]. This structure leads naturally to several vectorization and parallelization strategies.

If we sweep through the computational domain along lines of constant $i + j$ (in 2D), each term along these lines is independent of the others and depends only on data that has already been updated during the current sweep. This type of fine grain parallelization is ideal for vector computers. However, that degree of parallelism is not efficient for parallel computers because the extra communication time between processors exchanging data more than offsets the gain in computational efficiency. To optimize this algorithm for a parallel architecture, we need to break up the computational domain into blocks and send each block to a different processor. At the boundaries between the blocks, we reduce the data dependency between the blocks by using data from the previous time step along the block boundaries. This effectively reduces those points into a Jacobi iteration. However, the interior points are still solved with a Gauss-Seidel iteration. As long as the blocks are large enough that there are many more interior points than boundary points, then the overall convergence rate is approximately the same as Gauss-Seidel.

2.2.4 Approximate Riemann Solver

The fluxes on the right hand side of eqn(20) are discretized using a Roe-type approximate Riemann solver.[8] In this method the overall solution is built upon the solutions to the Riemann problem defined by the discontinuous jump in the solution between each pair of cells. The numerical flux for a first-order accurate (in space) Roe-type solver is written in symmetric form

as

$$F_{i+1/2} = \frac{1}{2}\left(F_{i+1} + F_i\right) - \frac{1}{2}\sum_k l_k \left(Q_{j+1} - Q_j\right) |\lambda_k| \, r_k \tag{30}$$

where r_k is the k^{th} right eigenvector, λ_k is the absolute value of the k^{th} eigenvalue, and l_k is the k^{th} left eigenvector. The values at the cell interface $(i+1/2)$ are obtained by a simple average of the neighboring cells. These first order accurate upwind fluxes are used in the vicinity of sharp discontinuities in order to suppress oscillations in the solution. We achieve a globally second order accurate solution by using a flux limiter that modifies the first order flux so that it uses second order central differencing in smooth portions of the flow. We are using a minmod limiter.[9]

Once the eigenvalues and eigenvectors are obtained and properly normalized to avoid singularities, it is relatively straight-forward to apply this scheme to the one-dimensional ideal MHD equations.[10,11] Unlike for the Euler equations, the extension to more than one dimension is non-trivial. The reason is that in more than one dimension, the Q vector must include B_x in addition to the other magnetic field components. (For the one-dimensional case B_x is constant by virtue of $\nabla \cdot \mathbf{B} = 0$). Since the $\mathbf{j} \times \mathbf{B}$ force acts perpendicularly to the directions of $\mathbf{j}$ and $\mathbf{B}$, the F flux vector has a zero term corresponding to B_x. Thus, the Jacobian matrix of F is singular and has a zero eigenvalue. This means we no longer have a complete set of physically meaningful eigenvectors. Physically, we expect information to travel either at the fluid velocity or at the fluid velocity plus or minus the wave speeds. Simply dropping B_x from the equation set is not a viable option, because B_x needs to change in order to maintain $\nabla \cdot \mathbf{B} = 0$. Powell et al., recently solved this problem by modifying the Jacobian in such a way as to change the zero eigenvalue to v_x (keeping the others unchanged), and then adding in a source term that exactly cancelled out the terms introduced by the modified Jacobian.[12]

The source term is

$$S_{div} = -\begin{bmatrix} \rho \\ \mathbf{B} \\ \mathbf{v} \\ \mathbf{v} \cdot \mathbf{B} \end{bmatrix} \nabla \cdot \mathbf{B} \tag{31}$$

It is proportional to the divergence of $\mathbf{B}$ and thus very small.

To this point we have only considered the hyperbolic terms. For reasonably large values of Re and Rm (easily in the range of interest for most

applications), the parabolic terms can be differenced explicitly without constraining the allowable time step. In this work we difference the parabolic terms explicitly in time with central differences in space. They are added to the right hand side fluxes arising from the approximate Riemann solver.

3 Accomplishments

3.1 Approximate Riemann Solver for Ideal MHD

3.1.1 1D Coplanar MHD (Magnetic Shock Tube)

The first test case we ran was a coplanar MHD Riemann problem. The coplanar MHD equations are obtained from the full one-dimensional ideal MHD equations by setting B_z and v_z to zero, thus allowing only planar flow and fields. This eliminates the $v_x \pm V_{Ax}$ eigenvalues, leaving a system of five equations with five eigenvalues. Mathematically, the Riemann problem is an initial boundary value problem in which there is initially a discontinuous jump in the data such that the left half of the domain is at one state and the right half of the domain is at another state. As the solution evolves in time, shock waves and rarefaction waves form that travel at speeds related to the wave speeds of the system. Although not physically realizable in plasmas, this problem is analogous to a shock tube in hydrodynamics. This test problem serves as validation of the approximate Riemann solver, because the computed solution can be compared to the exact analytical solution.

For our test case we used the same initial conditions as in Brio and Wu in order to allow direct comparison with their published results.[10] The initial left state was $p = 1$, $\rho = 1$, and $B_y = 1$. The initial right state was $p = 0.1$, $\rho = 0.125$, and $B_y = -1$. The velocities were zero and B_x was 0.75. Figure 1 shows the initial density distribution and its numerical solution after 400 time steps on an 800 point grid with a CFL number of 0.8. Figure 2 is the corresponding plot of the transverse magnetic field (B_y). The solution was computed using explicit time stepping since the pseudo time-stepping has not yet been implemented. The solution clearly shows five waves formed corresponding to the five eigenvalues. They are a fast rarefaction wave, a slow shock, a contact surface moving to the right, a slow compound wave (rarefaction and shock), and a fast rarefaction wave moving to the left. Note that the numerical method is able to resolve the shocks over a few grid points without introducing numerical oscillations. This is one of the advantages of the flux splitting approach we have used. The computed solution overlaid

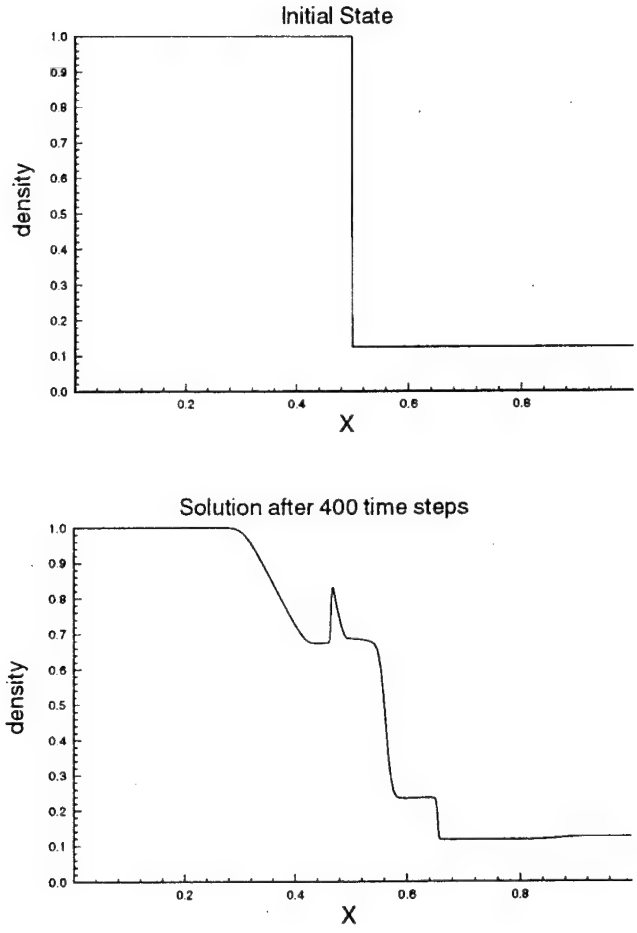

Figure 1: *Numerical solution of coplanar Riemann problem. Density profile is shown initially and after solution has evolved for 400 time steps.*

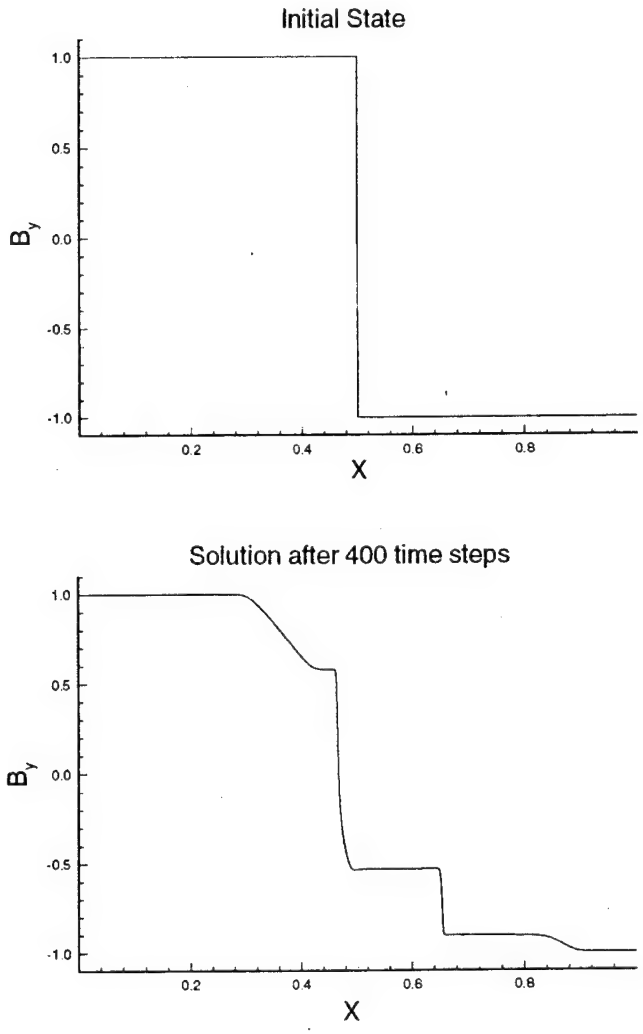

Figure 2: *Numerical solution of coplanar Riemann problem. Transverse magnetic field profile is shown initially and after solution has evolved for 400 time steps.*

exactly on Brio and Wu's published solution.

3.1.2 2D Test Cases

The next series of tests were done to validate the approximate Riemann solver in two dimensions to ensure that the zero eigenvalue fix was correctly implemented. The extra dimension allowed us to solve non-trivial steady-state problems in order to validate the LU-SGS implicit relaxation scheme as well.

The geometry for these tests consisted of two parallel plates with the velocities initialized to zero between the plates. The plates were perfectly conducting, so the component of the magnetic field normal to the plate was held at zero. A horizontal velocity and horizontal or vertical magnetic field were then imposed on the left boundary and the solution was run out to the steady-state. In the first test we initialized $v_x = 3$ on the left boundary and a uniform $B_x = 0.2$ throughout the domain. Since the field was parallel to the flow, there was no force exerted by the magnetic field, so the solution evolved to uniform horizontal flow with the initial magnetic field throughout the channel, as expected. When the solution converged to near machine zero, the divergence of the magnetic field was below 10^{-14} throughout the domain. This same steady-state result was achieved regardless of whether explicit time stepping or the LU-SGS implicit scheme was used.

The next test was the same except that we set $B_y = 0.2$ on the left boundary with no initial field in the domain. For this problem we used a relatively coarse grid with 20 cells in the y direction and 40 cells in the x direction. In this case, as the vertical field was convected into the domain, the perfectly conducting walls forced the field to zero at the walls, resulting in curvature of the field lines near the walls and a magnetic force on the plasma. For the relatively weak field in this case, the field was not enough to overcome the inertia of the plasma and was pushed out of the domain. Again, the divergence of the magnetic field gives a good indication that the zero eigenvalue fix has been correctly implemented. Figure 3 is a plot of the field divergence along a vertical slice. The divergence at the boundaries is slightly larger than in the interior because the inherent numerical viscosity causes the change in field at the wall to be smeared over more than one cell. The overall value is kept at an acceptably small value.

This two-dimensional steady-state solution was obtained with explicit time stepping at a CFL number of 0.8 and with the LU-SGS implicit relaxation scheme at an infinite CFL number (approximate Newton iteration).

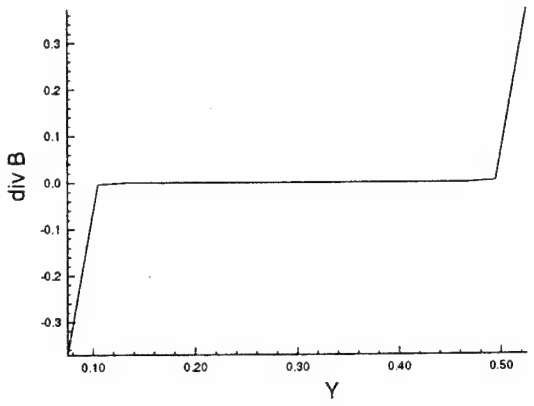

Figure 3: *Divergence of the magnetic field along a vertical slice for a steady-state solution for flow through a horizontal 2D channel with perfectly conducting walls and a vertical magnetic field and horizontal velocity imposed at the left boundary.*

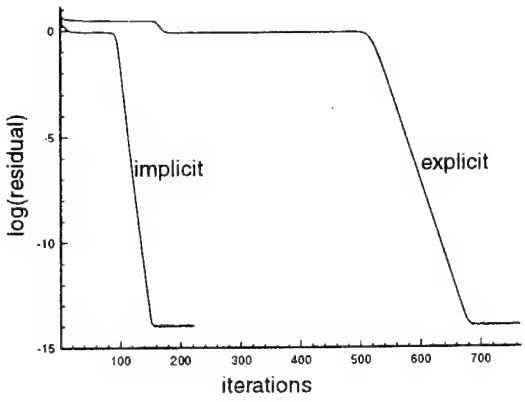

Figure 4: *Logarithm of the two-norm of the energy equation residual per cell plotted as a function of iteration number for explicit and implicit solutions of channel flow with horizontal velocity and vertical magnetic field imposed at the left boundary.*

Figure 4 is a plot of the logarithm of the two-norm of the residual per cell of the energy equation as a function of the number of iterations. The implicit scheme converged to 10^{-14} in about 150 iterations, whereas the explicit scheme required about 700 iterations. This is an acceleration factor of about 4.5 for the implicit scheme. Higher acceleration factors can be achieved for finer grids.

3.2 Viscous and Resistive MHD

The viscous and resistive terms in the MHD equations comprise the right hand side of the equality in eqn(1). The addition of these terms to the algorithm involved the modification of the R vector in eqn(20).

$$R \rightarrow R + \nabla \cdot \bar{\bar{T}}_p \tag{32}$$

The R vector is updated with each iteration to produce a solution that is fully coupled.

Using the divergence form of the parabolic terms reduces the differencing errors of the method. To preserve the accuracy on irregular meshes the derivatives are computed using a finite volume method.

The validation of the parabolic terms consisted of applying the code to a suite of test problems with known analytical solutions. We validated independently the terms associated with viscosity and those associated with resistivity and then the combined effect of all of the terms. The test problems were: (1) fully developed laminar flow between two parallel plates, (2) magnetic field generated by a constant current density, and (3) Hartmann flow. All of these test problems were run until a steady state solution developed.

3.2.1 Laminar Flow

We benchmarked the code to two types of laminar flows between infinite parallel plates. The plates restrict the flow to be one dimensional. No magnetic fields are present. This reduces the MHD equations to the Navier-Stokes equations. In these simulations a no-slip boundary condition was applied to the fluid in contact with the plates.

The first type of flow to which we benchmarked was viscous flow generated by one plate moving relative to the other plate. With no pressure gradient, constant viscosity, and incompressible flow, the equations reduce

to

$$(Re)^{-1} \nabla \cdot \bar{\bar{\tau}} = (Re)^{-1} \nabla^2 v_x = 0 \tag{33}$$

which is Laplace's equation. For finite viscosity (Re) the analytical solution for the flow velocity is

$$v_x(y) = V_0 \left(1 - \frac{y}{L}\right) + V_L \frac{y}{L} \tag{34}$$

where V_0 is the velocity of the plate at $y = 0$ and V_L is the velocity of the plate at $y = L$.

The errors between the analytical solution and the code generated solution were below 10^{-9} (the two-norm of the error between the solutions). We performed the same simulation with no viscosity ($Re \to \infty$). As would be expected, the flow velocity vanished everywhere except on the plates.

The next test was laminar flow between stationary parallel plates with a constant pressure gradient in the flow direction. The governing equation is

$$\nabla_x \cdot \left(p\bar{\bar{\mathrm{I}}}\right) = \frac{\partial p}{\partial x} = (Re)^{-1} \nabla^2 v_x. \tag{35}$$

The solution for this flow is the parabolic equation given by

$$v_x(y) = (Re) \frac{\partial p}{\partial x} \left(\frac{y^2 - 2y}{2}\right). \tag{36}$$

Figure 5 shows the solution generated by the code. Again the errors were reduced to below 10^{-9}.

3.2.2 Resistive Diffusion

We benchmarked the resistive diffusion to a current sheet with a uniform current density. Values of the tangential magnetic field were specified at parallel infinite plates, in a similar way as the first of the laminar flow simulations.

For no flow velocity and constant resistivity the MHD equations reduce to a Laplace equation similar to eqn(33).

$$(Rm)^{-1} \nabla \cdot \nabla \mathbf{B} = 0 \tag{37}$$

This equation has the same form for its solution as eqn(34).

$$B_x(y) = B_0 \left(1 - \frac{y}{L}\right) + B_L \frac{y}{L} \tag{38}$$

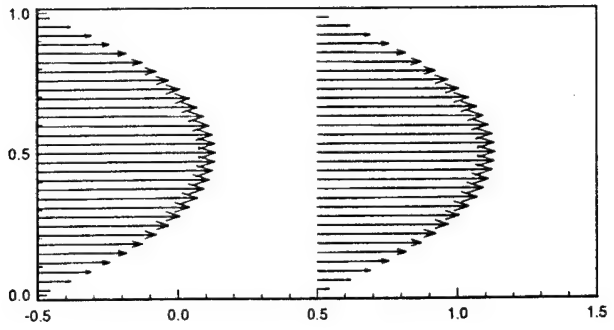

Figure 5: *Simulation of laminar flow between parallel plates in the presense of a constant pressure gradient. The velocity profile is parabolic as expected from the analytical solution.*

where B_0 is the velocity of the plate at $y = 0$ and B_L is the velocity of the plate at $y = L$.

The code agreed with the analytical solution to within errors of 10^{-9}.

3.2.3 Hartmann Flow

Hartmann flow combines the effects of viscosity and resistivity. The problem geometry is the same as that for the laminar flow with the addition of a magnetic field that is normal to the plates, in the y direction. See Figure 6 for an illustration.

The governing equations for the Hartmann flow can be found by combining the magnetic field and momentum equations from the MHD model. As before there will only be flow in the x direction. However, an applied electric field in the z direction must be included since it can generate an $\mathbf{E_o} \times \mathbf{B_o}$ flow in the x direction. The Hartmann flow is described by the differential equation,

$$\frac{\partial^2 v_x}{\partial y^2} - \frac{H^2}{L^2}\left(v_x + \frac{E_o}{B_o}\right) = 0, \tag{39}$$

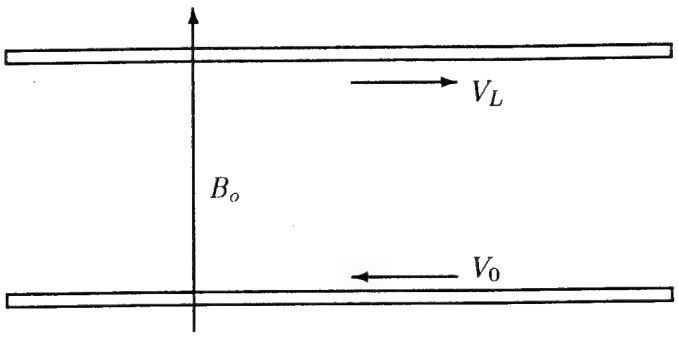

Figure 6: *The Hartmann flow geometry showing the moving parallel plates and the cross magnetic field.*

where the Hartmann number is defined as

$$H \equiv \frac{B_o L}{\sqrt{\rho \nu \eta}} = Al L \sqrt{Re Rm}. \tag{40}$$

The analytical solution to the Hartmann flow is

$$\begin{aligned} v_x(y) = V_0 \frac{\sinh\left(H(1-y/L)\right)}{\sinh(H)} + V_L \frac{\sinh(Hy/L)}{\sinh(H)} \\ -\frac{E_o}{B_o}\left[1 - \frac{\sinh\left(H(1-y/L)\right) + \sinh(Hy/L)}{\sinh(H)}\right] \end{aligned} \tag{41}$$

where the same no-slip boundary conditions have been applied. In the limit of no magnetic field, the solution reduces to the laminar flow solution, eqn(34).

The response of the magnetic field can be determined by solving the magnetic field equation for the field component that will be "dragged" with the flow. This magnetic field is described by

$$\frac{\partial B_x}{\partial y} = -(Rm)\left(v_x + \frac{E_o}{B_o}\right). \tag{42}$$

Using the flow solution of eqn(41), the solution for B_x is

$$B_x(y) = \left(\frac{Rm}{H}\right)\left(\frac{V_L - V_0}{2}\right)\left[\frac{\cosh(H/2) - \cosh\left(H(L-2y)/2L\right)}{\sinh(H/2)}\right]. \tag{43}$$

The boundary conditions are that B_x vanishes at the plates and the net current is zero. The first boundary condition may seem arbitrary, but it is

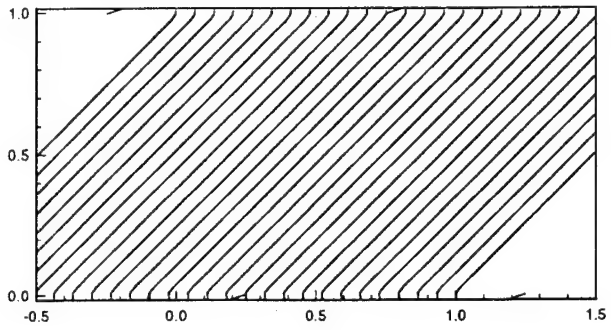

Figure 7: *Hartmann flow simulation with* $H = 10000$. *Flow velocity vectors and magnetic field lines are shown. The velocity of the flow is zero everywhere except at the plates. The magnetic field lines have a constant slope through the domain.*

consistent with the no-slip boundary condition applied to the flow solution. The second boundary condition relates the applied electric field, E_o, and the plate velocities.

$$\frac{E_o}{B_o} = -\frac{V_0 + V_L}{2} \tag{44}$$

Since the MHD equation set does not allow for an applied electric field, V_0 is set to $-V_L$, so that $E_o = 0$.

We performed simulations for large, small, and intermediate Hartmann numbers, H.

For a large Hartmann number, the effects of viscosity and resistivity are small, and the solution approaches that of ideal MHD. The flow velocity vanishes everywhere except on the plates, like it does for the inviscid case ($Re \rightarrow \infty$). The magnetic field is frozen into the plates and develops a slope (constant B_x) as the plates move. The slope of the magnetic field is determined by the value of H (the field lines slip through the plates due to resistivity). The slope of the magnetic field lines (B_o/B_x) is constant at H/Rm. Figure 7 shows the results from simulation with $H = 10000$. A finite value of the flow velocity exists only at the plates. The magnetic field lines are straight except at the plates where B_x is forced to vanish because

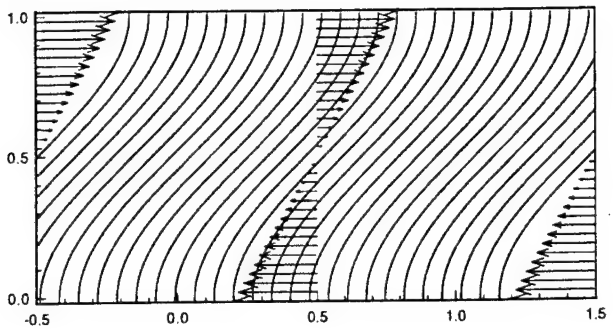

Figure 8: *Hartmann flow simulation with $H = 0.1$. Flow velocity vectors and magnetic field lines are shown. The velocity profile is linear and the magnetic field lines have an "S" shape caused by the bulk fluid flow.*

of the boundary conditions. For clarity the slope of the magnetic field has been normalized to unity at the midplane between the plates for all of the Hartmann flow simulations.

The limiting case of small Hartmann number is characterized by a flow that is dominated by viscous effects and a magnetic field that responds to the bulk fluid flow and the large resistivity. The flow velocity varies linearly from the velocity of the top plate to the velocity of the bottom plate, as described by eqn(34). The magnetic field diffuses through the plate and the bulk fluid, but the fluid drags the field lines along with the flow. This produces a swayed "S" shape to the field lines with a peak magnetic field at the midplane. Since the slope of the field lines is inversely proportional to the magnitude of B_x, the peak in the magnetic field corresponds to the field lines with the minimum slope (most horizontal). The minimum slope is $4/Rm$. The simulation results for $H = 0.1$ are shown in Figure 8. Notice the linear velocity profile and the swayed magnetic field lines.

Flows with Hartmann numbers in the intermediate ranges have solutions which exhibit characteristics of both of the limiting cases. The flow velocity falls to zero away from the boundaries in a scale length of L/H. This scale length is an appreciable fraction of the domain. The magnetic field is influenced by the motion of the plates and the fluid flow. The magnetic field

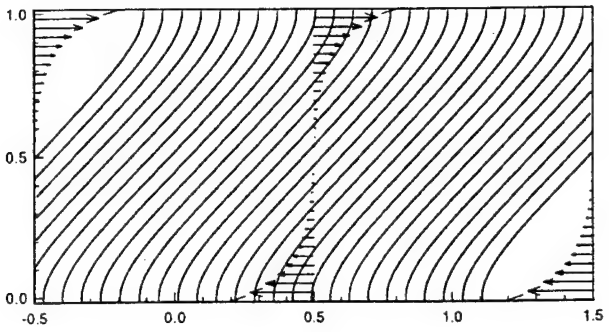

Figure 9: *Hartmann flow simulation with $H = 10$. Flow velocity vectors and magnetic field lines are shown. The flow velocity only exists close to the plates. The magnetic field lines are linear around the midplane.*

has a swayed shape close to the plates and is linear around the midplane. Away from the boundaries ($L/H < y < L - L/H$), the value of B_x is constant at $B_o Rm/H$. Figure 9 shows the results from a simulation with $H = 10$. The velocity profile falls to zero around the midplane. The magnetic field lines have a swayed shape like those in Figure 8 but not as dramatic, and they are linear around the midplane.

All of the Hartmann flow simulations converged to the analytical solution to within errors of 10^{-6}.

3.3 Parallel LU-SGS for Euler Model

A major goal of this project is to develop a 3D computational MHD algorithm which can feasibly solve problems on very large grids (a million elements or more). This goal can only be achieved with the use of parallel computers.

In preparation for the development of the full three-dimensional LU-SGS algorithm, we began to investigate strategies for implementing the algorithm on parallel architectures. We did this using a two-dimensional version of the algorithm applied to the Euler equations, a subset of the MHD equations. This section describes our first and the simplest approach to parallel imple-

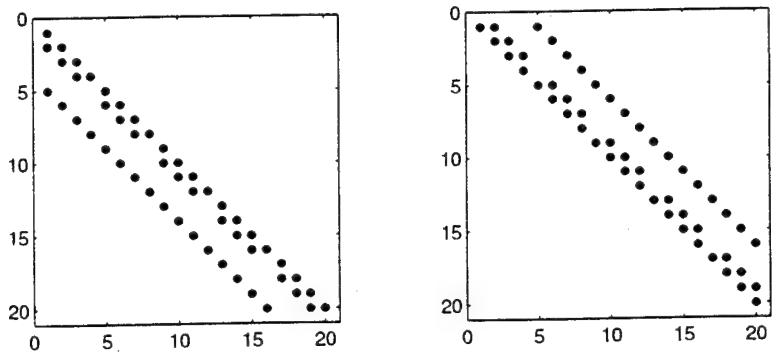

Figure 10: *The* 20×20 *lower (a) and upper (b) tridiagonal block matrices for the LU-SGS algorithm with a grid of* 4×5 *cells.*

mentation.

The LU-SGS algorithm involves a double sweep of the computational domain. This is a predictor-corrector method. The forward (predictor) sweep solves a lower tridiagonal block matrix for the entire computational domain. The backward (corrector) sweep solves an upper tridiagonal block matrix.

Figure 10 shows the form of the lower and upper block diagonal matrices for the case of a 4×5 grid. The first sub (super) diagonal of the lower (upper) matrix is interspersed with zeros at regular intervals — every $I_{max} - 1$ elements. Systems of equations that can be divided into lower and upper matrices are easily solved using backward and forward substitutions. Because the form of the matrices for this algorithm [eqn(29)], the solutions at grid cells with constant $i+j$ are independent. This means the algorithm vectorizes easily. While vectorization is easy to implement, it does not provide the scalability of parallelization.

3.3.1 Description of the Implementation

The simplest parallel implementation is to decompose the domain into its component cells, to distribute the grid cells over the processors of the parallel computer, and to treat each cell as residing on a different processor. This approach exploits the independencies of the solutions of the cells with constant $i + j$. Communication between the cells provides the necessary

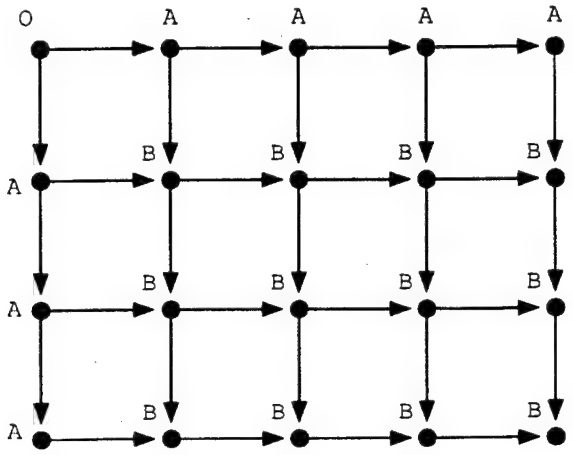

Figure 11: *Computational domain showing the data flow and communication from the cells of a* 4×5 *grid the forward sweep.*

synchronization.

Figure 11 shows the data flow and communication from the cells of a 4×5 grid during the forward sweep. The grid cells can be classified as one of three types based on their data dependencies. Type O cells of which there is only one have no data dependencies. Type A cells only depend on one cell which is upstream from the sweep direction. Type B cells depend on two upstream cells. Processors with cells of type A or B do not begin computing until the data from the upstream cells is received. The processor with the type O cell begins computing immediately. This data dependency provides the synchronization to prevent non-deterministic solutions. For the backward sweep the lower right cell of Figure 11 is the type O cell, and the sweep progresses toward the upper left corner of the domain.

For our implementation, we used the Parallel Virtual Machine (PVM) communication library which was developed at Oak Ridge National Laboratory.[13] PVM allowed us to connect a network of four DEC Alpha workstation and use them as our parallel computer. XPVM, the graphical interface for PVM, proved to be a valuable tool during debugging.

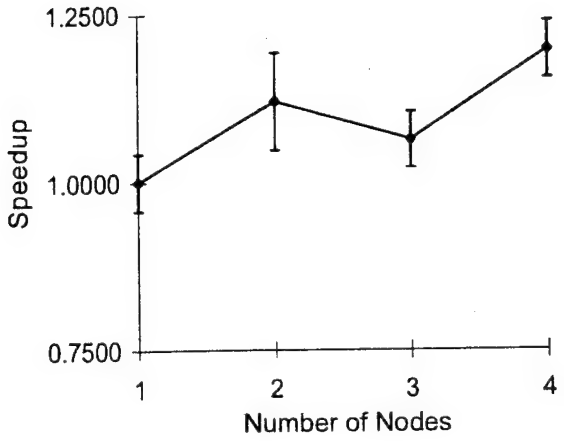

Figure 12: *The speedup measured for a grid of constant size as more processors are used. The error bars indicate the standard deviation over several measurements.*

3.3.2 Results

The two measures that we used for determining parallel effectiveness are (1) the speedup obtained when a problem grid of constant size is evenly distributed onto more processors and (2) the speedup obtained when the problem grid is increased proportional to the number of processors. Speedup is defined as the time required to find the solution with n processors divided by the time with one processor. For perfectly parallel implementations, the speedup for the first measure will be equal to the number of processors, and the speedup for the second measure will be unity. Any communication time or processor synchronization will decrease the speedup.

To measure the speedup for a constant problem grid, we used a 4×4 grid and varied the number of processors from one to four. Since there are four variables per cell for the 2D Euler equations, this grid produced a 64×64 matrix. While this was a small size problem, it was sufficient to test the parallel implementation. The speedup results are shown in Figure 12. Some speedup can be seen; however, the amount is unsatisfactory. The same data is presented in Figure 13 showing the fractional parallel efficiency. Ideally the efficiency would remain close to unity.

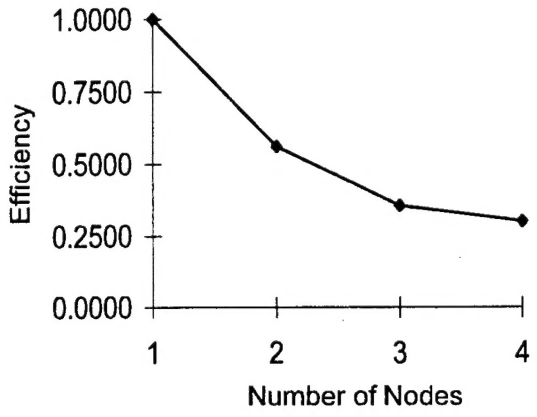

Figure 13: *The fractional parallel efficiency measured for a grid of constant size as more processors are used.*

The measure of speedup for constant size grid per processor revealed that the amount of computation time increased proportional to the size of the grid (number of processors).

These low efficiencies indicate that the simplest approach for parallel implementation of the LU-SGS algorithm is inadequate. These results are not surprising since the grain of parallelization in this approach is too fine and requires excessive communication. The number of grid cells in practical applications will be much greater than the number of processors. This suggests dividing the domain into a number of large blocks, so that the grid cells within a block are located on the same processor (and memory) and do not need to communicate through message passing. The only message passing that occurs is the communication between cells on the boundary surfaces of differing blocks. This coarse-grain parallelization holds the most promise and is the next approach we will implement. The best situation would be to have a parallel machine that is composed of vector processors, so that both fine-grain vectorization and coarse-grain parallelization could be exploited.

4 Professional Interactions

4.1 Project Personnel

The personnel directly involved in this project are listed below.

Name	Position
Dr. Uri Shumlak	Research Assistant Professor
Dr. D. Scott Eberhardt	Associate Professor
Dr. Thomas R. Jarboe	Professor
Mr. Ogden S. Jones	Graduate Student
Mr. Bogdan Udrea	Graduate Student
Mr. David Taflin	Graduate Student
Mr. Byoungsoo Kim	Graduate Student

4.2 Collaborations

4.2.1 Air Force Phillips Laboratory

Dr. Robert Peterkin and Dr. Thomas Hussey of the High Energy Plasma Physics Division on parallelization approaches to MACH3 and on the 3-D Rayleigh-Taylor instability in solid liners.

4.2.2 University of Michigan

Prof. Bram van Leer, Prof. Kenneth Powell, and Prof. Philip Roe of the Aerospace Engineering Department on approximate Riemann solvers for the MHD plasma model and the zero eigenvalue issue.

4.2.3 University of Colorado

Prof. Steve McCormick of the Applied Math Department on three-dimensional multigrid algorithms.

4.2.4 University of Washington

Prof. Randy LeVeque of the Applied Math Department on approximate Riemann solvers and their applications to multidimensional problems.

4.3 Publications

Presently a journal article is being prepared for submission to the Journal of Computational Physics. The title is "An Approximate Riemann-type

Algorithm for Resistive/Viscous Magnetohydrodynamics" by O. S. Jones, U. Shumlak, and D. S. Eberhardt.

4.4 Presentations

A paper explaining our project was presented at the Twenty-First Annual IEEE International Conference on Plasma Sciences, Santa Fe, New Mexico, June 1994. The title was "An Implicit Algorithm for the Ideal MHD Equations."

Two papers will be presented at the Seventh Joint EPS - APS International Conference on Physics Computing, Pittsburgh, Pennsylvania, June 1995. The titles are "A Fully Coupled MHD Solution for the Hall Effect Thruster" and "An Implicit Approximate Riemann Solver for Multi-Dimensional MHD Computations on Parallel Computers."

5 Conclusions

The successful development of the two-dimensional, viscous, resistive version of the advanced implicit algorithm and the initial implementation of the algorithm on parallel architectures indicate that we are making significant progress toward our project objectives. This research project has been presented at an international conference, and more presentations are planned. A journal article is being prepared for submission to a refereed journal.

Valuable collaborations have been formed with the Air Force Phillips Laboratory and other universities.

The continuing development of this project will include exploiting the parallel structure of the algorithm, extending the algorithm to three dimensions, and installing the three dimensional algorithm into Phillips Laboratory's MHD code, MACH3.

References

[1] U. Shumlak, T. W. Hussey, and R. E. Peterkin, Jr., IEEE Transaction on Plasma Science, 23, (1995).

[2] B. A. Nelson, T. R. Jarboe, D. J. Orvis, L. McCullough, J. Xie, C. Zhang, and L. Zhou, Phys. Rev. Lett., 72, 3666 (1994).

[3] F. M. Lehr, A. Alaniz, J. D. Beason, L. C. Carswell, J. H. Degnan, J. F. Crawford, S. E. Englert, T. J. Englert, J. M. Gahl, J. H. Holmes, T. W. Hussey, G. F. Kiuttu, B. W. Mullins, R. E. Peterkin, Jr., N. F. Roderick, P. J. Turchi, and J. D. Graham, J. Appl. Phys., 75, 3769 (1994).

[4] J. H. Degnan, R. E. Peterkin, Jr., G. P. Baca, J. D. Beason, D. E. Bell, M. E. Dearborn, D. Dietz, M. R. Douglas, S. E. Englert, T. J. Englert, K. E. Hackett, J. H. Holmes, T. W. Hussey, G. F. Kiuttu, F. M. Lehr, G. J. Marklin, B. W. Mullins, D. W. Price, N. F. Roderick, E. L. Ruden, C. R. Sovinec, P. J. Turchi, G. Bird, S. K. Coffey, S. W. Seiler, Y. G. Chen, D. Gale, J. D. Graham, M. Scott, and W. Sommars, Phys. Fluid B, 5, 2938 (1993).

[5] H. P. Furth, J. Kileen, and M. N. Rosenbluth, Phys. Fluids, 6, 479 (1963).

[6] H. Ok and D. S. Eberhardt, "Solution of Unsteady Incompressible Navier-Stokes Equations Using an LU Decomposition Scheme," AIAA-91-1611 (1991).

[7] S. Yoon and A. Jameson, AIAA J., 26, 1025 (1988).

[8] P. L. Roe, J. Computational Phys., 43, 357 (1981).

[9] R. J. LeVeque, *Numerical Methods for Conservation Laws,* Birkhauser Verlag, Boston (1992).

[10] M. Brio and C. C. Wu, J. Comp. Phys., 75, 400 (1988).

[11] A. L. Zachery and P. Colella, J. Comp. Phys., 99, 341 (1992).

[12] K. G. Powell, B. van Leer, and P. L. Roe, *Private Communication,* 1994.

[13] Al Geist, A. Beguelin, J. Dongara, W. Jiang, R. Manchek, V. Sunderam, *PVM 3 User's Guide and Reference Manual,* 1994.